AWS Customer Management System - Pricing Analysis

Executive Summary

This document provides a comprehensive cost analysis for the Customer Information Management System built on AWS serverless architecture. The system utilizes AWS Lambda, API Gateway, and DynamoDB to provide a scalable, cost-effective solution for managing customer data.

Key Cost Drivers: - AWS Lambda: Function execution and requests - API Gateway: REST API requests - DynamoDB: On-demand read/write operations and storage

Architecture Overview

The system consists of: - **5 Lambda Functions**: CreateCustomer, GetCustomers, GetCustomer, UpdateCustomer, DeleteCustomer - **1 API Gateway**: REST API with CORS enabled - **1 DynamoDB Table**: On-demand billing mode with customer records

Pricing Components Analysis

1. AWS Lambda Pricing

Request Pricing: - Standard requests: $0.0000002 per request - ARM requests: $0.0000002 per request (same price)

Compute Pricing (x86): - Tier 1 (0-6B GB-seconds): $0.0000166667 per GB-second - Tier 2 (6B-15B GB-seconds): $0.0000150000 per GB-second - Tier 3 (15B+ GB-seconds): $0.0000133334 per GB-second

Compute Pricing (ARM - Graviton2): - Tier 1 (0-7.5B GB-seconds): $0.0000133334 per GB-second - Tier 2 (7.5B-18.75B GB-seconds): $0.0000120001 per GB-second - Tier 3 (18.75B+ GB-seconds): $0.0000106667 per GB-second

2. API Gateway Pricing

REST API Requests: - First 333 million requests/month: $0.0000035 per request - Next 667 million requests/month: $0.0000028 per request - Next 19 billion requests/month: $0.0000023800 per request - Over 20 billion requests/month: $0.0000015100 per request

HTTP API Requests (Alternative): - First 300 million requests/month: $0.0000010 per request - Over 300 million requests/month: $0.0000009 per request

3. DynamoDB Pricing

On-Demand Throughput: - Read Request Units: $0.125 per million RRUs - Write Request Units: $0.625 per million WRUs

Storage: - First 25 GB/month: Free - Beyond 25 GB: $0.25 per GB/month

Free Tier Benefits: - 25 GB storage per month (always free) - 2.5 million DynamoDB Streams read requests per month

Cost Scenarios

Scenario 1: Low Usage (Startup/Development)

Monthly Usage: - API Requests: 10,000 - Lambda Invocations: 50,000 (10 per API call) - DynamoDB Operations: 30,000 reads, 20,000 writes - Data Storage: 5 GB

Cost Breakdown: - **Lambda Requests**: 50,000 × $0.0000002 = $0.01 - **Lambda Compute**: 50,000 × 0.5s × 0.128GB × $0.0000166667 = $0.05 - **API Gateway**: 10,000 × $0.0000035 = $0.035 - **DynamoDB Reads**: 30,000 × $0.000000125 = $0.004 - **DynamoDB Writes**: 20,000 × $0.000000625 = $0.013 - **DynamoDB Storage**: 5 GB (Free tier)

Total Monthly Cost: $0.10

Scenario 2: Medium Usage (Small Business)

Monthly Usage: - API Requests: 100,000 - Lambda Invocations: 500,000 - DynamoDB Operations: 300,000 reads, 200,000 writes - Data Storage: 50 GB

Cost Breakdown: - **Lambda Requests**: 500,000 × $0.0000002 = $0.10 - **Lambda Compute**: 500,000 × 0.5s × 0.128GB × $0.0000166667 = $0.53 - **API Gateway**: 100,000 × $0.0000035 = $0.35 - **DynamoDB Reads**: 300,000 × $0.000000125 = $0.038 - **DynamoDB Writes**: 200,000 × $0.000000625 = $0.125 - **DynamoDB Storage**: (50-25) × $0.25 = $6.25

Total Monthly Cost: $7.39

Scenario 3: High Usage (Enterprise)

Monthly Usage: - API Requests: 1,000,000 - Lambda Invocations: 5,000,000 - DynamoDB Operations: 3,000,000 reads, 2,000,000 writes - Data Storage: 200 GB

Cost Breakdown: - **Lambda Requests**: 5,000,000 × $0.0000002 = $1.00 - **Lambda Compute**: 5,000,000 × 0.5s × 0.128GB × $0.0000166667 = $5.33 - **API Gateway**: 1,000,000 × $0.0000035 = $3.50 - **DynamoDB Reads**: 3,000,000 × $0.000000125 = $0.375 - **DynamoDB Writes**: 2,000,000 × $0.000000625 = $1.25 - **DynamoDB Storage**: (200-25) × $0.25 = $43.75

Total Monthly Cost: $55.21

Cost Optimization Recommendations

1. Immediate Optimizations

- **Use ARM-based Lambda functions**: 20% cost savings on compute
- **Optimize Lambda memory allocation**: Right-size memory for actual usage
- **Implement response caching**: Reduce redundant API calls
- **Use DynamoDB efficiently**: Batch operations where possible

2. Architecture Optimizations

- **Consider HTTP API instead of REST API**: 71% cost reduction for API Gateway
- **Implement connection pooling**: Reduce Lambda cold starts
- **Use DynamoDB single-table design**: Optimize for access patterns
- **Enable DynamoDB auto-scaling**: For predictable workloads

3. Monitoring and Alerting

- **Set up CloudWatch billing alarms**: Monitor unexpected cost increases
- **Use AWS Cost Explorer**: Analyze spending patterns
- **Implement usage tracking**: Monitor per-customer costs
- **Regular cost reviews**: Monthly optimization assessments

Free Tier Benefits

AWS Lambda: - 1 million requests per month - 400,000 GB-seconds of compute time per month

API Gateway: - 1 million REST API calls per month (first 12 months)

DynamoDB: - 25 GB storage (always free) - 25 read capacity units and 25 write capacity units (always free) - 2.5 million stream read requests per month (always free)

Annual Cost Projections

Scenario	Monthly Cost	Annual Cost	With Optimizations
Low Usage	$0.10	$1.20	$0.96 (20% savings)
Medium Usage	$7.39	$88.68	$70.94 (20% savings)
High Usage	$55.21	$662.52	$530.02 (20% savings)

Risk Factors and Considerations

Cost Risks

- **Unexpected traffic spikes**: On-demand pricing can scale costs rapidly
- **Data growth**: Storage costs increase linearly with data volume
- **API abuse**: Uncontrolled API access can drive up costs

Mitigation Strategies

- **Implement rate limiting**: Control API request rates
- **Set up billing alerts**: Early warning for cost overruns
- **Use reserved capacity**: For predictable workloads
- **Implement data lifecycle policies**: Archive old data

Conclusion

The Customer Management System provides excellent cost efficiency for small to medium workloads, with costs starting as low as $0.10/month for development environments. The serverless architecture ensures you only pay for actual usage, making it ideal for variable workloads.

Key Takeaways: - Very low entry cost with generous free tiers - Linear scaling with usage patterns - Significant optimization opportunities available - Predictable cost structure for planning

Next Steps: 1. Implement ARM-based Lambda functions for immediate 20% savings 2. Set up comprehensive monitoring and alerting 3. Consider HTTP API migration for high-traffic scenarios 4. Regular monthly cost optimization reviews

---

Analysis Date: November 22, 2025
Region: US East (N. Virginia)
Currency: USD